Assignment 4: The SLOB SLAB

CS 444: Operating Systems II

Fall 2016

Abstract

This project involves researching and implementing a SLAB memory manager with best fit algorithms. The goal is to gain an understanding of how Memory Management works inside of an operating system. Through work our process of learning about and implementing the SLAB is shown.

Joshua D. Bowen
Chris J. Mendez

November 27, 2016

Assignment 4, CS444 Christopher Mendez, Joshua Bowen

1 MM Driver Design

Our first decision was to research the already existing SLOB. We found in the SLOB the already existent Memory Allocation system and then the memory placement portion. The SLOB already had a system in place to place or create pages at the first location that it could find space. The modification we needed to make is one that takes this but instead of placing at the first spot, it instead finds the best spot.

To implement this design we first came up with a mental model of what this looked like.

The SLOB would iterate accross the list of pages looking for the best fit. When the current position is a better fit than the current best fit, the new best fit is remembered. Once the SLOB reaches the end of the list it will place the item at the bes fit location. If no fit is found at all, then a new page is generated to fit the item.

Using this mental model we knew what it was that we needed to do. With this in mind we sketched up some pseudocode.

2 Questions

2.1 What do you think the main point of this assignment is?

The main point of this assignment was to learn more about the various aspects within the kernel of the Operating Systems. In this specific assignment we were supposed to be able to look at the SLOB and find what it was that needed to be changed. Then as in previous assignments we were responsible for getting it working with the rest of the OS. We continued with the tradition of having some strange error or bug in the code that we needed to flex our problem solving skills to solve.

On top of the figuring out how to get the SLAB working, we were also tasked with how to test it. I think one of the points of this assignment was to once again make sure that we were able to understand our assignment and make tests that thoroughly tested the code. In general these assignments often have different ways of testing the algorithm that is different from every other part of the kernel.

2.2 How did you personally approach the problem? Design decisions, algorithm, etc.

The first step, of course, was to research the already existing SLOB. After looking into the SLOB we were able to determine what would need to be changed. In general the SLOB already had a functioning Memory Allocation system which didn't need to be changed. What we needed to change was how how the SLOB determined where pages went. Our goal is to change the "First Find" fit to a "Best Fit".

We decided that all we needed to change was this aspect of the SLOB. We then wrote up a design for the algorithm before implementing. This design can be seen above.

After crafting our design we began implementation. After implementation came testing.

2.3 How did you ensure your solution was correct? Testing details, for instance.

Originally, the plan was to modify the data structure struct page, and to include an element that would be the difference between the requested allotment and the allotment available. This was a way overengineered way of approaching the problem, and one that Im not sure would work. As

soon as we began programming and tried modifying the data structure, there was a huge amount of compile errors. This made us feel like we were on the wrong track and so rethought the approach a bit. From what we understood of the current slob allocation, the slob_page_alloc function handled finding where to place a block and placing it. The first fit algorithm would just check if there was a difference between the allocated request and page, and if there was, it could place the requested blocks there. We rewrote that so that when it got to that point where there was a difference it would keep track of the best fit, and then upon going through all the pages, allocate the blocks at the spot with the best fit.

In order to test this, we created two system calls. The first kept track of the memory being claimed by the slob file. It did this by starting at 0 and every time a page was allocated, it increased by the page size. There was also the memory being used. The memory being used would just increment and decrement as blocks were allocated. The first system call returned the memory claimed the second returned the memory claimed - the memory being used. By dividing the difference in claimed and used over the total claimed, we would be able to see the fragmentation.

2.4 What did you learn?

We learned about the SLOB and memory management in general. We also learned how to debug and test a different aspect of the kernel. Additionally, we learned how to familiarze ourselves with another aspect of the kernel and make changes to what we needed so that it would do what we wanted.

3 History

3.1 Git Version Control Log

Commit	Message
commit 741bdb5031847e5c06c69bcd794508a7a4644efc Author: bowenjos ¡bowenjos@oregonstate.edu¿ Date: Sun Nov 27 21:33:41 2016 -0800	Write up Update and Testing
commit c5e1486fee8df91f4d0c97e591640f0a5a6f1db5 Author: bowenjos ¡bowenjos@oregonstate.edu¿ Date: Sun Nov 20 12:06:10 2016 -0800	pdf test
commit a13f36ede9488602d97ec180cbee421c8a2cf2d2 Author: bowenjos ¡bowenjos@oregonstate.edu¿ Date: Sun Nov 20 11:51:07 2016 -0800	Some kernel stuff, some write up stuff
commit 9960f2a450f6dd8f87a93ed7481d9678e6fdeec5 Author: bowenjos ¡bowenjos@oregonstate.edu¿ Date: Sun Nov 20 11:16:47 2016 -0800	Added write up files

3.2 Work Log

Date	Time	What
11/15/2016	3pm-5m	Began researching the currently implemented SLOB and gaining and understanding of how the memory management works.
11/19/2016	10am-12pm	Began some code.
11/20/2016	10am-12pm	Continued working on the assignment, began writing our formal design document.
11/23/2016	8am-12pm	Tested implementation.
11/27/2016	9pm-10pm	Finished Write up.